FUNDAÇÃO HERMÍNIO OMETTO

Sistemas de Informação

Projeto Interdisciplinar I / 3ºB/2025

Tecnologias Inovadoras para promover Energia Limpa, Sustentável e Acessível

[Empresa de energia solar com um mercado de créditos]

[Bugig Solar]

116983 / Enzo Corrêa Caetano

114928 / Felipe Claudiano da Silva

116459 / Harry Felipe Santos Souza

114477 / Lucas Eduardo de Campos

116544 / Renan dos Santos Alvares

1ª Fase do Projeto Técnico

Orientações:

Entregar este documento preenchido no **SchoolNet** até o prazo definido no cronograma do Projeto Interdisciplinar, com os requisitos solicitados inseridos no próprio documento e **Anexos** em arquivos separados.

1 Descrição da Proposta do Projeto

Um sistema de mercado de créditos.

O nosso software apresentará os dados das economias geradas e um sistema de mercado de créditos, onde contará com compras e vendas.

Requisitos sistema:

O sistema deve permitir que os usuários (Cliente e Fornecedor) realizem cadastro.
O sistema deve permitir que os usuários façam login com credenciais válidas.

O sistema deve validar as credenciais de login.

O sistema deve permitir que o usuário se identifique como Cliente (Comprador) ou

Fornecedor (Vendedor).

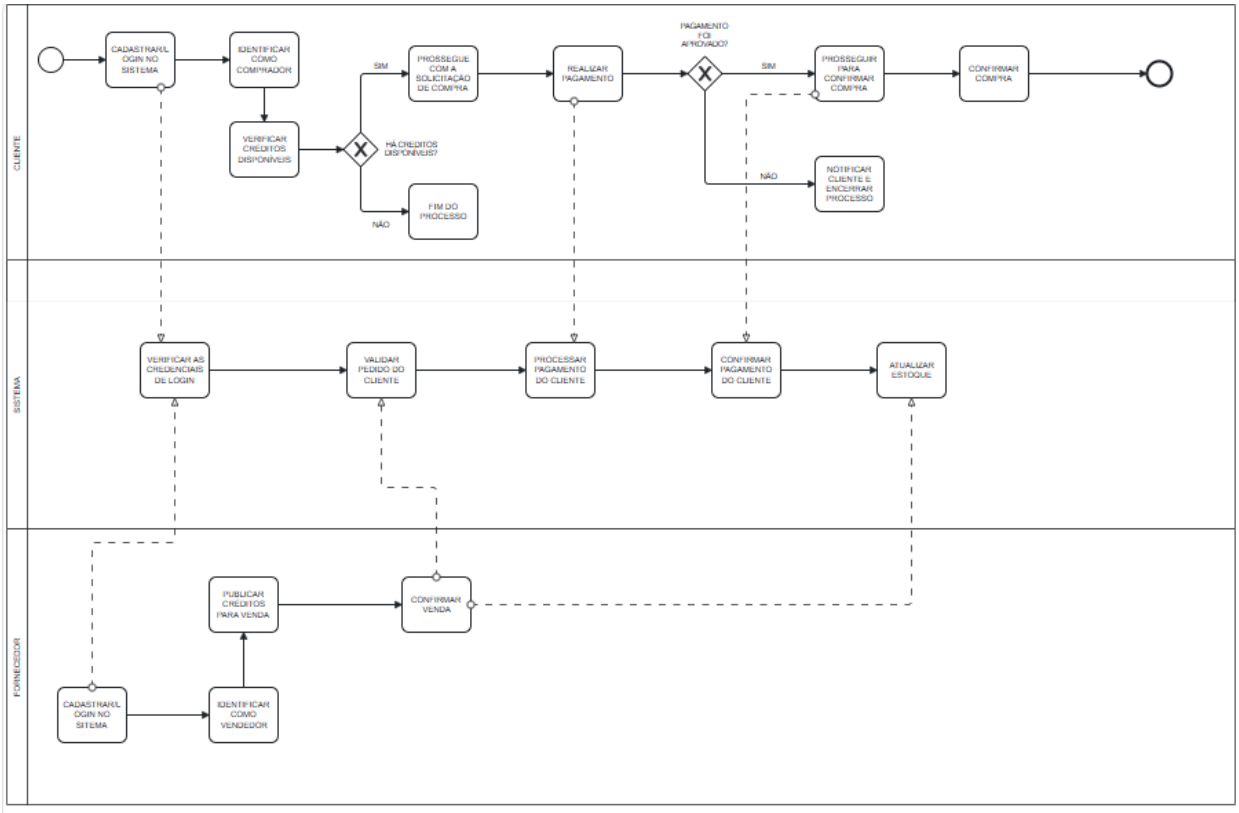
O sistema deve permitir que o fornecedor publique créditos para venda.

O sistema deve atualizar o estoque de créditos após cada venda.

O sistema deve exibir os créditos disponíveis para compra.
O sistema deve permitir que o cliente solicite uma compra.
O sistema deve processar o pagamento da compra.
O sistema deve validar se o pagamento foi aprovado.
O sistema deve notificar o cliente caso o pagamento não seja aprovado.
O sistema deve confirmar a compra após o pagamento ser aprovado.

O sistema deve apresentar um relatório apresentando os resultados de crédito.
O sistema deve permitir que os usuários encerrem suas sessões

2 Modelagem de Negócios



O sistema permite a compra e venda de créditos de energia solar, proporcionando uma plataforma eficiente para compradores e fornecedores. O Cliente acessa o sistema para adquirir créditos, iniciando pelo cadastro ou login. Em seguida, define seu perfil como comprador e pode verificar as ofertas disponíveis. Após escolher a oferta desejada, ele solicita a compra e realiza o pagamento correspondente. Com o pagamento efetuado, confirma a transação para garantir que foi concluída corretamente. Ao finalizar suas operações, o Cliente pode encerrar a sessão.

Por outro lado, o Fornecedor utiliza o sistema para disponibilizar créditos de energia solar para venda. O processo se inicia com o cadastro ou login, seguido da definição do perfil como vendedor. Ele então publica os créditos, estabelecendo a quantidade disponível e o valor correspondente. O Fornecedor também pode conferir as vendas realizadas, garantindo que os créditos foram vendidos e os pagamentos processados corretamente. Ao concluir suas atividades, ele pode encerrar a sessão.

O Sistema opera automaticamente para garantir o funcionamento adequado do mercado de créditos de energia solar. Ele valida os pedidos dos clientes, verifica a disponibilidade dos

créditos ofertados, processa os pagamentos realizados, confirma as transações e atualiza o estoque de créditos, assegurando que as vendas sejam refletidas corretamente no sistema.

3 Programação Orientada a Objetos

1. Classe Interface:

Esta classe atua como a "porta de entrada" do sistema, controlando a interação direta com o usuário através do terminal.

Ela é responsável por apresentar as opções disponíveis, receber as entradas do usuário e exibir as respostas do sistema de forma organizada.

Gerenciar o processo de login, autenticando os usuários.

Apresentar menus claros e intuitivos, facilitando a navegação entre as diferentes funcionalidades do sistema.

Exibir mensagens informativas, como confirmações de transações, mensagens de erro e resultados de consultas.

Coordena a comunicação com as outras classes do sistema, solicitando e exibindo informações conforme necessário.

2. Classe Cadastro Usuário:

Esta classe é responsável por gerenciar o processo de cadastro de novos usuários no sistema.

Ela coleta e armazena as informações dos usuários.

Solicitar e receber os dados dos usuários, como nome, e-mail e documento, por exemplo.

Armazenar os dados dos clientes no banco de dados.

3. Classe Fornecedor:

Esta classe gerencia as ações dos usuários que atuam como fornecedores de créditos.

Ela permite que os fornecedores vendam seus créditos para outros usuários.

Verificar o saldo de créditos disponíveis do fornecedor.

Permitir que o fornecedor defina o valor dos créditos que deseja vender.

4. Classe Comprador:

Esta classe gerencia as ações dos usuários que atuam como compradores de créditos.

Ela permite que os compradores adquiram créditos de outros usuários.

Permitir que o comprador defina o valor dos créditos que deseja comprar.

Listar os fornecedores disponíveis e valores de créditos.

Registrar as transações de compra de créditos, atualizando os créditos comprados.

5. Classe Venda Créditos:

Esta classe é responsável por gerenciar as transações de venda de créditos.

Atualizar os saldos dos compradores e vendedores após cada transação.

6. Classe Compra Créditos:

Esta classe é responsável por gerenciar as transações de compra de créditos.

Atualizar os saldos dos compradores e vendedores após cada transação.

7. Classe Consulta Banco de Dados:

Esta classe centraliza o acesso ao banco de dados, fornecendo métodos para realizar consultas e manipulações de dados.

Fornece métodos para realizar consultas personalizadas ao banco de dados.

Otimizar o desempenho das consultas ao banco de dados.

8 - Classe Relatório:

imprimir no terminal o relatório de valor de credito disponível, quantidade vendido se for vendedor e quantidade comprada se for comprador do usuário que está acessando o sistema.

Ela armazena informações detalhadas sobre quantidade comprada. ex: valor.

Ela armazena informações detalhadas sobre quantidade vendida. ex: valor.

Gerar relatórios de vendas.

Registrar as informações de cada compra e venda de créditos.

Gerar relatórios de compras.

4 Banco de Dados

1. Fornecedor

Descrição: Entidade que oferece créditos para os compradores.

Os atributos base são: id, nome , e-mail, endereço, CPF, senha de acesso ao sistema e total vendido.

2. Comprador

Descrição: Entidade que adquire créditos de fornecedores.

Os atributos são semelhantes ao do fornecedor: id, nome, e-mail, CPF, endereço, senha de acesso ao sistema e total comprado.

3. Crédito

Descrição: Entidade que representa os créditos negociados entre fornecedores e compradores.

Os atributos são: id, valor do crédito, data de emissão, data de vencimento e o id do fornecedor.

4. Pagamento

Descrição: Entidade que registra os pagamentos realizados pelos compradores para liquidar créditos adquiridos.

Os atributos são: id, id do crédito, valor pago e data de pagamento

5. Compra

Descrição: Entidade que mantém o histórico completo de todas as transações, incluindo compras, vendas e pagamentos.

Os atributos são: id, id do crédito, quantidade, id do fornecedor, id do comprador, data de transação e valor da transação

6. Endereço

Descrição: Entidade que mantém as localizações armazenadas no sistema.

Os atributos são: id, rua, cidade, sigla do estado.

7. Fornecedor_endereço

Descrição: Vincula um endereço a um fornecedor

Os atributos são: id, id do endereço e id do fornecedor

8. Cliente_endereco

Descrição: Vincula um endereço a um cliente

Os atributos são: id, id do endereço e id do cliente

5 Considerações Finais

Desenvolver um trabalho em grupo é um desafio de todos pensarem a mesma coisa e se unirem para torna-la funcional. No banco de dados as principais dificuldades foram criar as tabelas e relaciona-las, já na POO, foi a conciliação das classes com seus atributos e métodos, além de especificar suas ações e torna-las funcionais.

Referências Bibliográficas

[Elenque as referências bibliográficas utilizadas neste documento, seguindo as normas da ABNT (ver abaixo) e em ordem alfabética por sobrenome do primeiro autor.]

EXEMPLOS DE REFERÊNCIA DE LIVRO:

TANENBAUM, Adrew S. **Sistemas operacionais modernos**. 3. ed. São Paulo: Pearson Prentice Hall, 2010.

EXEMPLOS DE REFERÊNCIA DE ARTIGO PUBLICADO EM REVISTA CIENTÍFICA:

SOARES, Edira Pólido do Carmo; DUARTE, Marina Dantas de Oliveira; ALMEIDA, Adiel Teixeira de. Planejamento de Sistemas de Informação baseado na metodologia BSP: um estudo do caso DETRAN/AL. **Sistemas & Gestão**, v. 3, n. 3, p. 163-177, 2009.

EXEMPLOS DE REFERÊNCIA DE ARTIGO PUBLICADO EM ANAIS DE CONGRESSO:

ANTONELLO, Sérgio Luis; CARDOSO, Rogério. Olimpíada de Raciocínio Lógico: relatos de uma competição para alunos ingressantes em curso de nível superior. In: WORKSHOPS DO CONGRESSO BRASILEIRO DE INFORMÁTICA NA EDUCAÇÃO, 2015. Maceió. **Anais...** 2015. p. 1263-1270.

EXEMPLOS DE REFERÊNCIA ELETRÔNICA (SITES, PDFs):

LEFFA, Vilson José. **Normas da ABNT:** citações e referências bibliográficas. 2016. Disponível em: <http://www.leffa.pro.br/textos/abnt.htm>. Acesso em: 5 abr 2016.

FHO-UNIARARAS. **Balanço social**. 2014. Disponível em: <http://uniararas.br/download.php?file=FHO_BS2014.pdf>. Acesso em: 15 mai 2016.

EXEMPLOS DE SOFTWARE:

ADOBE SYSTEMS. Adobe acrobate 5.0. San Jose, 2001. Disponível em: <http://www.adobe.com>. Acesso em: 10 mai 2002.

EXEMPLOS DE REFERÊNCIA DE TESE DE DOUTORADO:

ANTONELLO, Sergio Luís. **Um sistema de planejamento e gestão para bacias hidrográficas com uso de análise multicritérios**. 2008. Tese (Doutorado em Ecologia de Agroecossistemas) - Escola Superior de Agricultura “Luiz de Queiroz”, Universidade de São Paulo, Piracicaba, 2008.

EXEMPLOS DE REFERÊNCIA DE DISSERTAÇÃO DE MESTRADO:

PERUCCI, Camilo César. **Método de aplicação de lean thinking para desenvolvimento de projetos de software om metologia SCRUM**. 2016. Dissertação (Mestrado) – Universidade Metodista de Piracicaba, Engenharia de Produção, Santa Bárbara D’oeste, 2016.

EXEMPLOS DE REFERÊNCIA DE TCC:

ZANCHETA, Fernando Eduardo. **Desenvolvimento de um sistema de apoio a decisão para uma instituição de ensino superior**. 2016. Trabalho de Conclusão de Curso (Bacharelado em Sistemas de Informação) - FHO, Araras. 2016.

EXEMPLOS DE AUTOR CORPORATIVO:

PIRACICABA. Secretaria Municipal de Planejamento. **Perfil socioeconômico do município de Piracicaba**. Piracicaba: Prefeitura do Município de Piracicaba, 1998. 68 p.

EXEMPLOS DE AUTOR DE CAPÍTULO DE LIVRO:

LEHMAN, H. Environmental ethics and pesticide use. In: PIMENTEL, D. (Ed.). **Techniques for reducing pesticide use**: economic and environmental benefits. Chichester: John Wiley, 1997. cap. 3, p. 35-50.

EXEMPLOS DE DOCUMENTO JURÍDICO:

BRASIL. Constituição (1988). **Constituição da República Federativa do Brasil**. Brasília: Senado, 1988. 168 p.

EXEMPLOS DE CITAÇÃO DE CONTEÚDO NO TEXTO:

um autor

Conteúdo / afirmação ... (CERVO, 1976).

um autor incluindo a página (este formato vale também para os casos seguintes)

Conteúdo / afirmação ... (CERVO, 1976, p.25).

dois (ou três) autores

Conteúdo / afirmação ... (CERVO; BERVIAN, 1978).

mais de três autores (usar o "et al."):

Conteúdo / afirmação ... (BASTOS et al., 1979).

EXEMPLO DE CITAÇÃO "EM LINHA" NO TEXTO:

um autor

O trabalho de Cervo (1978) relaciona o processo de ...

dois (ou três) autores

O trabalho de Cervo e Bervian (1978) relaciona o processo de ...

mais de três autores (usar o "et al."):

Segundo Bastos et al. (1979), a estratégia de ...

uso do apud (embora seja conveniente evitar):

Segundo Silva (1983 apud ABREU, 1999, p.3), o problema ocorre porque ...

EXEMPLO DE CITAÇÃO DIRETA COM ATÉ 3 LINHAS:

Segundo Saraiva e Fiori (2017, p.24): "Os professores Camilo e Fabiano são os melhores professores do curso de Sistemas de Informação da UNIARARAS."

2ª Fase do Projeto Técnico

Orientações:

Entregar este documento preenchido no **SchoolNet** até o prazo definido no cronograma do Projeto Interdisciplinar, com os requisitos solicitados inseridos no próprio documento e **Anexos** em arquivos separados.

1 Descrição da Proposta do Projeto

O BugigSolar é um sistema voltado para a comercialização de créditos de energia solar entre usuários. A plataforma permitirá que usuários cadastrados atuem como Clientes (Compradores) ou Fornecedores (Vendedores) de créditos, oferecendo funcionalidades de compra, venda, gerenciamento de transações e relatórios.

Requisitos do sistema:

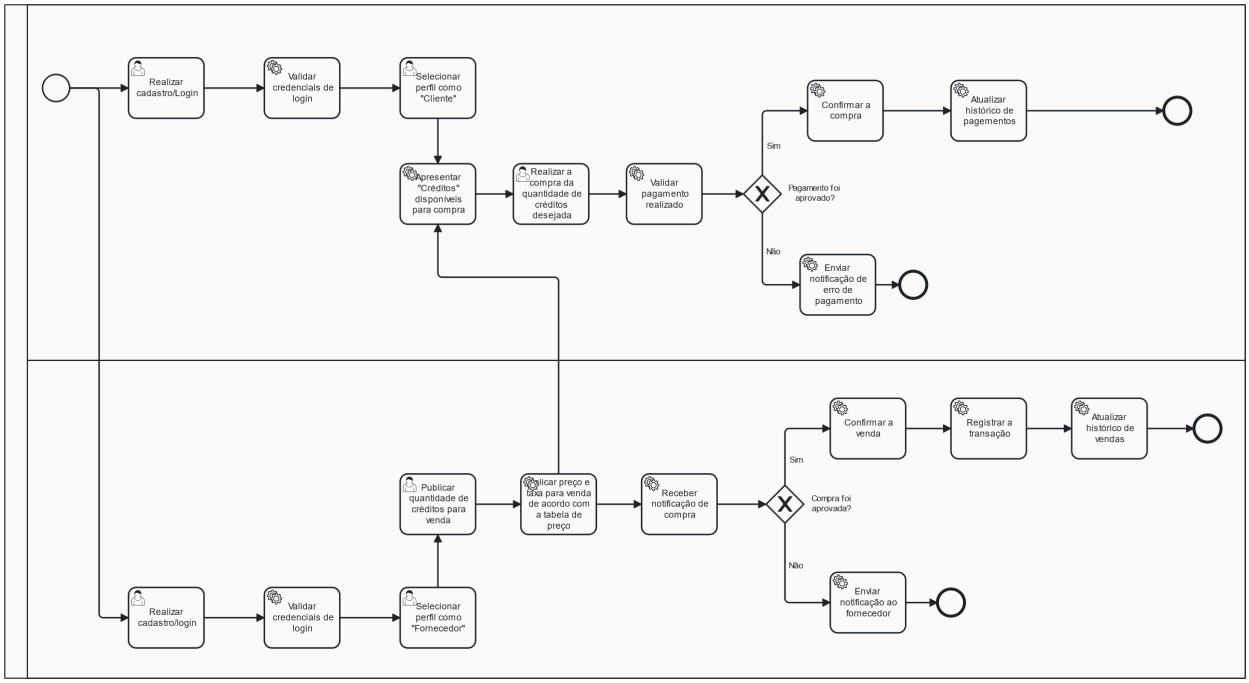
- *Cadastro e Autenticação*
  - *O sistema deve permitir o cadastro de novos usuários (Clientes e Fornecedores).*
  - *O sistema deve permitir o login com credenciais válidas.*
  - *O sistema deve permitir que o usuário selecione seu perfil: Cliente ou Fornecedor.*
  - *O sistema deve permitir que o usuário encerre sua sessão (logout).*

- *Funcionalidades para Fornecedores*
  - *O sistema deve permitir que o fornecedor publique créditos disponíveis para venda.*
  - *O sistema deve aplicar uma taxa de venda com base na Unidade Federativa (UF) do fornecedor, conforme a tabela única de preços previamente cadastrada.*
  - *O sistema deve processar e registrar a venda realizada.*
  - *O sistema deve validar a quantidade de créditos disponíveis antes de concluir a venda.*
  - *O sistema deve confirmar a venda somente após a confirmação do pagamento pelo cliente.*

- *Funcionalidades para Clientes*
  - *O sistema deve permitir que o cliente selecione e compre créditos publicados.*
  - *O sistema deve processar o pagamento da compra.*
  - *O sistema deve validar a aprovação do pagamento.*
  - *O sistema deve notificar o cliente em caso de falha na aprovação do pagamento.*
  - *O sistema deve confirmar a compra após aprovação do pagamento.*

- *Histórico e Relatórios*
  - *O sistema deve manter um histórico de todas as transações realizadas.*
  - *O sistema deve registrar vendas realizadas pelos fornecedores.*
  - *O sistema deve registrar pagamentos realizados pelos clientes.*
  - *O sistema deve gerar relatórios com o resumo de compras, vendas e pagamentos.*

2 Modelagem de Negócios



O sistema permite a compra e venda de créditos de energia solar, proporcionando uma plataforma eficiente para compradores e fornecedores. O Cliente acessa o sistema para adquirir créditos, iniciando pelo cadastro ou login. Em seguida, define seu perfil como comprador e pode verificar as ofertas disponíveis. Após escolher a oferta desejada, ele solicita a compra e realiza o pagamento correspondente. Com o pagamento efetuado, confirma a transação para garantir que foi concluída corretamente. Além disso, o Cliente pode consultar um relatório com o resultado de suas vendas e compras, permitindo um melhor acompanhamento das suas operações. Ao finalizar suas atividades, o Cliente pode encerrar a sessão.

Por outro lado, o Fornecedor utiliza o sistema para disponibilizar créditos de energia solar para venda. O processo se inicia com o cadastro ou login, seguido da definição do perfil como vendedor. Ele então publica os créditos, estabelecendo a quantidade disponível e o valor correspondente. O Fornecedor também pode conferir as vendas realizadas, garantindo que os créditos foram vendidos e os pagamentos processados corretamente. Ao concluir suas atividades, ele pode encerrar a sessão.

O Sistema opera automaticamente para garantir o funcionamento adequado do mercado de créditos de energia solar. Ele valida os pedidos dos clientes, verifica a disponibilidade dos créditos ofertados, processa os pagamentos realizados, confirma as transações e atualiza o estoque de créditos, assegurando que as vendas sejam refletidas corretamente no sistema.

3 Programação Orientada a Objetos

1. Classe main

Responsabilidade: Controla a interação com o usuário pelo terminal.

Funcionalidades: Apresentar opções e menus, receber entradas do usuário e exibe respostas, exibir mensagens informativas e de erro e coordenar a comunicação entre as classes do sistema.

Atributos: cli e forn;

2. Classe Cadastro de Usuário

Responsabilidade: Gerencia o cadastro de novos usuários.

Funcionalidades: Coletar dados (nome, e-mail, documento) e armazenar informações no banco de dados.

Atributos: nomeCadastro, senhaCadastro e confirmaSenha.

3. Classe Login

Responsabilidade: Gerencia a entrada dos usuários ao sistema.

Funcionalidades: verificar se aquele usuário já existe no banco de dados para permitir a entrada do mesmo no sistema.

Atributos: respostaUsuario; nomeFornecedor; senhaFornecedor; confirmaSenha; nomeCliente; String senhaCliente;

3. Classe Fornecedor

Responsabilidade: Gerenciar ações dos fornecedores de créditos.

Funcionalidades: Permitir que fornecedores publiquem créditos à venda e define o valor dos créditos para venda.

Atributos: nome e creditosDisponiveis

4. Classe Comprador

Responsabilidade: Gerenciar ações dos compradores de créditos.

Funcionalidades: Permite a compra de créditos de fornecedores, lista fornecedores e valores de créditos disponíveis e registra transações de compra e atualiza créditos comprados.

Atributos: nome; id; creditosComprados;

5. Classe Mercado

Responsabilidade: Gerenciar transações de venda de créditos.

Funcionalidades: atualizar saldos de compradores e vendedores após transações

Atributos: valorComprado; confirmaCompra; valorVendido; confirmaVenda; Rvenda Rcompra;

6. Classe Consulta Banco de Dados

Responsabilidade: Centraliza o acesso ao banco de dados.

Funcionalidades: Realizar consultas e manipulações de dados.

7. Classe Relatório

Responsabilidade: Gerar relatórios de transações.

Funcionalidades: Imprimir relatórios de créditos disponíveis, quantidades compradas e vendidas e armazena detalhes sobre cada transação.

Atributos: creditosDisponiveis; historicoTransacoes; quantidadeVendidaPorFornecedor; quantidadeCompradaPorComprador;

4 Banco de Dados

1. Usuários

Descrição: Entidade que armazena os dados padrões dos usuários

Os atributos são: id, nome, e-mail, endereço, número, Documento, senha de acesso ao sistema

2. Fornecedores

Descrição: Entidade que oferece créditos para os compradores.

Os atributos base são: id, Razão Social, Telefone comercial e UF.

3. Clientes

Descrição: Entidade que adquire créditos de fornecedores.

Os atributos são: id, Telefone pessoal, Descrição.

4. Ofertas

Descrição: Entidade que representa os créditos negociados entre fornecedores e compradores.

Os atributos são: id, valor do crédito, data de emissão, data de vencimento e o id do fornecedor.

5. Pagamento

Descrição: Entidade que registra os pagamentos realizados pelos compradores para liquidar créditos adquiridos.

Os atributos são: id, id do crédito, valor pago e data de pagamento.

6. Vendas

Descrição: Entidade que mantém o histórico completo de todas as transações, incluindo compras, vendas e pagamentos.

Os atributos são: id do mercado, id do crédito, quantidade, id do fornecedor, id do comprador, data de transação e valor da transação

7. Tabela_preco

Descrição: padroniza os preços e taxa por UF

Os atributos são: id, UF, valor e taxa.

8. Tipo_usuario

Descrição: registra o tipo dos usuários.

Os atributos são: id, nome_tipo.

9. Tipo_pagamento

Descrição: Registra o tipo dos pagamentos disponíveis.

Os atributos são: id, nome_tipo.

5 Considerações Finais

Enfrentamos desafios na modelagem do banco de dados lógico, especialmente na definição das relações entre usuários, fornecedores e clientes, assim como na estruturação do sistema de precificação dos créditos. Em Programação Orientada a Objetos (POO), a principal dificuldade foi identificar corretamente os atributos e compreender seu funcionamento dentro do sistema. Já na modelagem de negócios, o maior obstáculo foi mapear todos os processos de forma precisa para refinar o modelo BPMN.

Referências Bibliográficas

[Elenque as referências bibliográficas utilizadas neste documento, seguindo as normas da ABNT e em ordem alfabética por sobrenome do primeiro autor.]

3ª Fase do Projeto Técnico

Orientações:

Entregar este documento preenchido no **SchoolNet** até o prazo definido no cronograma do Projeto Interdisciplinar, com os requisitos solicitados inseridos no próprio documento e **Anexos** em arquivos separados.

1 Descrição da Proposta do Projeto

O BugigSolar é um sistema voltado para a comercialização de créditos de energia solar entre usuários. A plataforma permitirá que usuários cadastrados atuem como Clientes (Compradores) ou Fornecedores (Vendedores) de créditos, oferecendo funcionalidades de compra, venda, gerenciamento de transações e relatórios.

Requisitos do sistema:

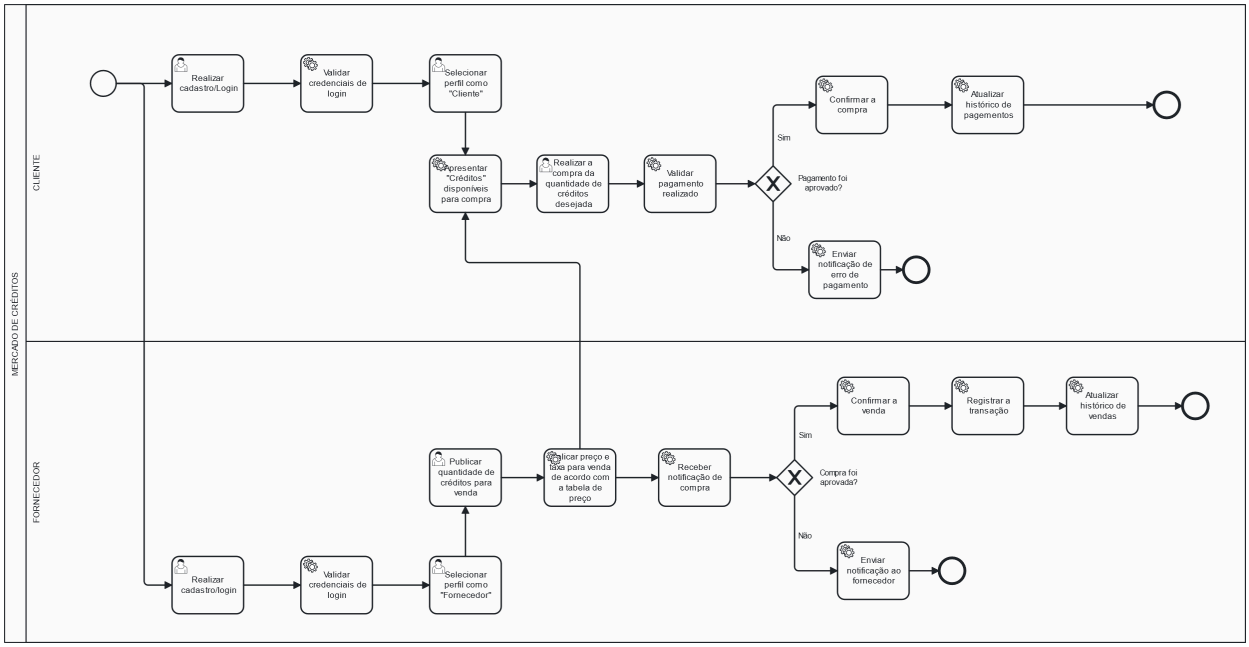
1. *Cadastro e Autenticação*
    a. *O sistema deve permitir o cadastro de novos usuários (Clientes e Fornecedores).*
    b. *O sistema deve permitir o login com credenciais válidas.*
    c. *O sistema deve permitir que o usuário selecione seu perfil: Cliente ou Fornecedor.*
    d. *O sistema deve permitir que o usuário encerre sua sessão (logout).*

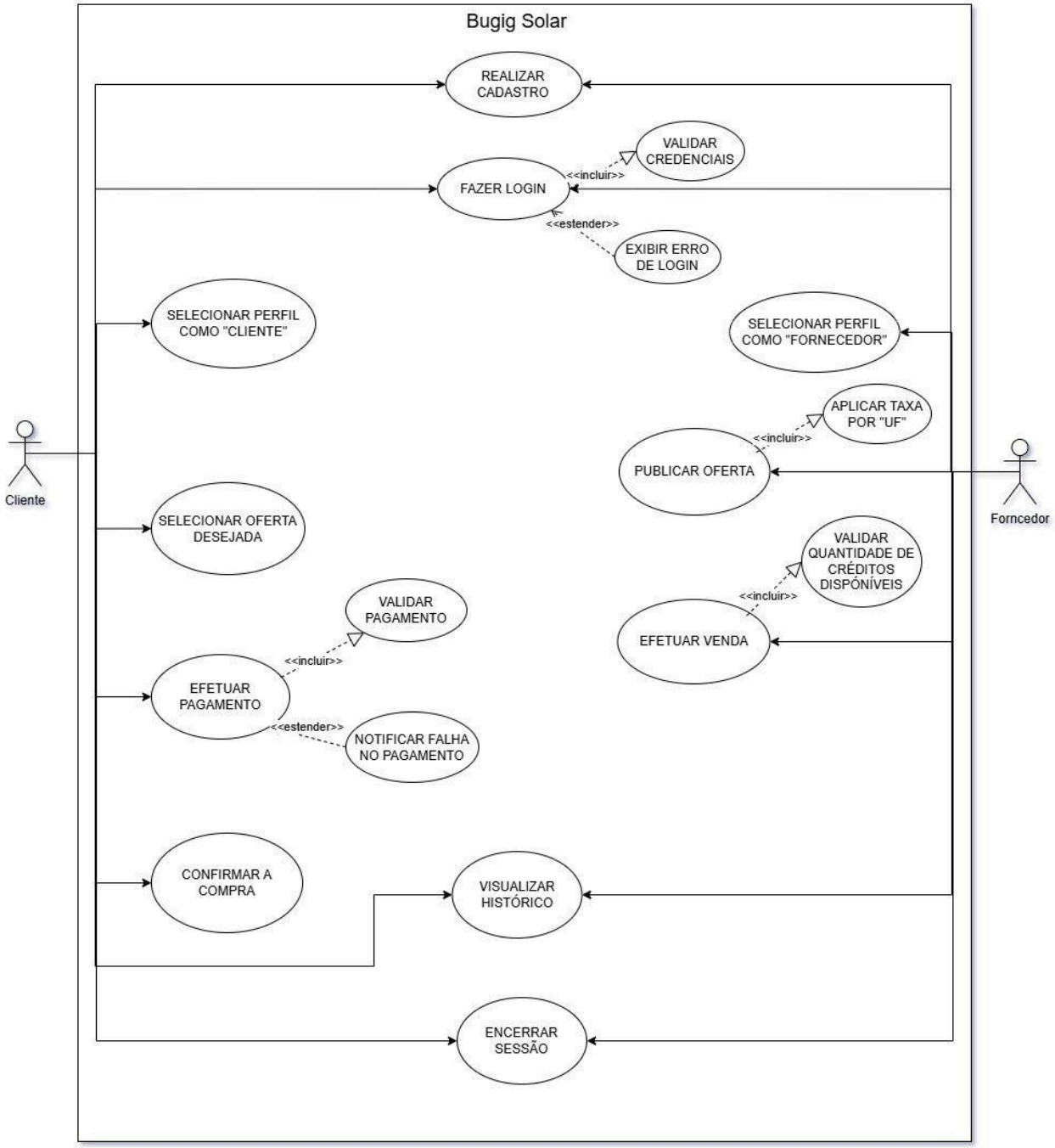
2. *Funcionalidades para Fornecedores*
    a. *O sistema deve permitir que o fornecedor publique créditos disponíveis para venda.*
    b. *O sistema deve aplicar uma taxa de venda com base na Unidade Federativa (UF) do fornecedor, conforme a tabela única de preços previamente cadastrada.*
    c. *O sistema deve processar e registrar a venda realizada.*
    d. *O sistema deve validar a quantidade de créditos disponíveis antes de concluir a venda.*
    e. *O sistema deve confirmar a venda somente após a confirmação do pagamento pelo cliente.*

3. *Funcionalidades para Clientes*
    a. *O sistema deve permitir que o cliente selecione e compre créditos publicados.*
    b. *O sistema deve processar o pagamento da compra.*
    c. *O sistema deve validar a aprovação do pagamento.*
    d. *O sistema deve notificar o cliente em caso de falha na aprovação do pagamento.*
    e. *O sistema deve confirmar a compra após aprovação do pagamento.*

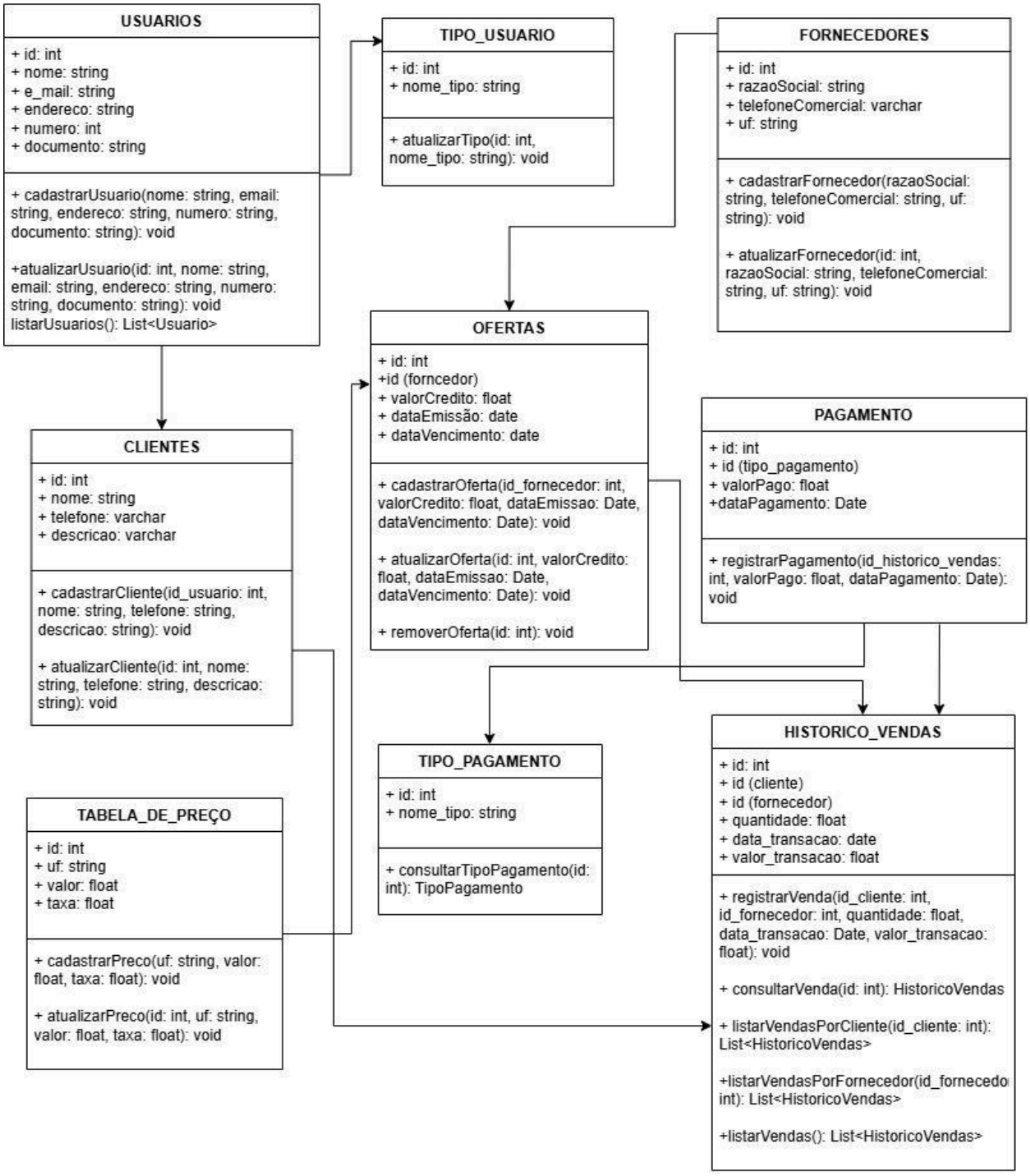
4. *Histórico e Relatórios*
    a. *O sistema deve manter um histórico de todas as transações realizadas.*
    b. *O sistema deve registrar vendas realizadas pelos fornecedores.*
    c. *O sistema deve registrar pagamentos realizados pelos clientes.*
    d. *O sistema deve gerar relatórios com o resumo de compras, vendas e pagamentos.*

Modelagem de Negócios




Bugig Solar
REALIZAR CADASTRO
VALIDAR CREDENCIAIS
<<incluir>>
FAZER LOGIN
<<estender>>
EXIBIR ERRO DE LOGIN
SELECIONAR PERFIL COMO "CLIENTE"
SELECIONAR PERFIL COMO "FORNECEDOR"
APLICAR TAXA POR "UF"
<<incluir>>
PUBLICAR OFERTA
Cliente
Forncedor
SELECIONAR OFERTA DESEJADA
VALIDAR QUANTIDADE DE CRÉDITOS DISPÓNÍVEIS
<<incluir>>
EFETUAR VENDA
VALIDAR PAGAMENTO
<<incluir>>
EFETUAR PAGAMENTO
<<estender>>
NOTIFICAR FALHA NO PAGAMENTO
CONFIRMAR A COMPRA
VISUALIZAR HISTÓRICO
ENCERRAR SESSÃO

BUGIG SOLAR


USUARIOS
+ id: int
+ nome: string
+ e_mail: string
+ endereco: string
+ numero: int
+ documento: string
+ cadastrarUsuario(nome: string, email: string, endereco: string, numero: string, documento: string): void
+atualizarUsuario(id: int, nome: string, email: string, endereco: string, numero: string, documento: string): void
listarUsuarios(): List<Usuario>
TIPO_USUARIO
+ id: int
+ nome_tipo: string
+ atualizarTipo(id: int, nome_tipo: string): void
FORNECEDORES
+ id: int
+ razaoSocial: string
+ telefoneComercial: varchar
+ uf: string
+ cadastrarFornecedor(razaoSocial: string, telefoneComercial: string, uf: string): void
+ atualizarFornecedor(id: int, razaoSocial: string, telefoneComercial: string, uf: string): void
OFERTAS
+ id: int
+id (forncedor)
+ valorCredito: float
+ dataEmissão: date
+ dataVencimento: date
+ cadastrarOferta(id_fornecedor: int, valorCredito: float, dataEmissao: Date, dataVencimento: Date): void
+ atualizarOferta(id: int, valorCredito: float, dataEmissao: Date, dataVencimento: Date): void
+ removerOferta(id: int): void
PAGAMENTO
+ id: int
+ id (tipo_pagamento)
+ valorPago: float
+dataPagamento: Date
+ registrarPagamento(id_historico_vendas: int, valorPago: float, dataPagamento: Date): void
CLIENTES
+ id: int
+ nome: string
+ telefone: varchar
+ descricao: varchar
+ cadastrarCliente(id_usuario: int, nome: string, telefone: string, descricao: string): void
+ atualizarCliente(id: int, nome: string, telefone: string, descricao: string): void
TIPO_PAGAMENTO
+ id: int
+ nome_tipo: string
+ consultarTipoPagamento(id: int): TipoPagamento
HISTORICO_VENDAS
+ id: int
+ id (cliente)
+ id (fornecedor)
+ quantidade: float
+ data_transacao: date
+ valor_transacao: float
+ registrarVenda(id_cliente: int, id_fornecedor: int, quantidade: float, data_transacao: Date, valor_transacao: float): void
+ consultarVenda(id: int): HistoricoVendas
+ listarVendasPorCliente(id_cliente: int): List<HistoricoVendas>
+listarVendasPorFornecedor(id_fornecedo int): List<HistoricoVendas>
+listarVendas(): List<HistoricoVendas>
TABELA_DE_PREÇO
+ id: int
+ uf: string
+ valor: float
+ taxa: float
+ cadastrarPreco(uf: string, valor: float, taxa: float): void
+ atualizarPreco(id: int, uf: string, valor: float, taxa: float): void

3 Programação Orientada a Objetos

1. Classe Main

Responsabilidade:
Gerenciar a interação com o usuário por meio do terminal.

Funcionalidades:

- Apresentar menus e opções disponíveis ao usuário.
- Receber entradas e comandos do usuário.
- Exibir mensagens informativas e mensagens de erro.
- Coordenar a comunicação entre as demais classes do sistema.

Atributos:

- respostaForCli
- respostaLogCad

---

2. Classe CadastroUsuario

Responsabilidade:
Gerenciar o processo de cadastro de novos usuários no sistema.

Funcionalidades:

- Coletar dados como nome, e-mail, documento, senha, cidade e estado.
- Validar e armazenar essas informações no banco de dados.

Atributos:

- nomeCadastro
- senhaCadastro
- confirmaSenha
- cpfCadastro
- cidadeCadastro

- emailCadastro
- estadoCadastro

Métodos:

- cadastroCliente() – Responsável por cadastrar um novo cliente.
- cadastroFornecedor() – Responsável por cadastrar um novo fornecedor.

---

3. Classe Login

Responsabilidade:
Controlar o acesso dos usuários ao sistema.

Funcionalidades:

- Validar as credenciais fornecidas pelo usuário.
- Verificar a existência do usuário no banco de dados.
- Permitir o acesso ao sistema conforme o tipo de usuário.

Atributos:

- respostaUsuario
- nomeFornecedor
- senhaFornecedor
- confirmaSenha
- nomeCliente
- senhaCliente

Métodos:

- loginFornecedor() – Valida o login de um fornecedor.
- loginCliente() – Valida o login de um cliente.

---

4. Classe Fornecedor

Responsabilidade:
Gerenciar as ações de fornecedores de créditos no sistema.

Funcionalidades:

- Publicar créditos disponíveis para venda.
- Definir o valor dos créditos ofertados.

Métodos:

- menuVendas() – Exibe a interface de venda e permite criar ofertas.

---

5. Classe Cliente

Responsabilidade:
Gerenciar as ações dos clientes que compram créditos.

Funcionalidades:

- Visualizar fornecedores e ofertas de créditos.
- Realizar compras de créditos disponíveis.
- Registrar transações e atualizar os créditos adquiridos.

Métodos:

- menuCompras() – Exibe as opções de compra e permite adquirir ofertas.

---

6. Classe Ofertas

Responsabilidade:
Gerenciar as ofertas criadas pelos fornecedores.

Funcionalidades:

- Publicar novas ofertas de créditos no sistema.

Atributos:

- valorVendido
- confirmaVenda
- rvenda

Método:

- vender() – Permite ao fornecedor registrar uma nova oferta.

---

7. Classe Compras

Responsabilidade:
Gerenciar o processo de compra efetuado pelos clientes.

Funcionalidades:

- Executar a transação de compra de créditos.

Atributos:

- valorComprado
- confirmaCompra
- rcompra

---

8. Classe ConsultaBancoDados

Responsabilidade:
Centralizar o acesso e manipulação de dados no banco de dados.

Funcionalidades:

- Executar consultas SQL.
- Realizar operações de inserção, atualização e exclusão de dados.

---

9. Classe Relatorio

Responsabilidade:
Gerar e apresentar relatórios de movimentações e transações do sistema.

Funcionalidades:

- Exibir créditos disponíveis.
- Listar quantidades de créditos comprados e vendidos.
- Armazenar e exibir detalhes sobre cada transação realizada.

Métodos:

- relatorioVendas()
- relatorioCompras()

4 Banco de Dados

1. Usuários

- **Descrição**: Entidade que armazena os dados principais dos usuários do sistema.
- **Atributos**:
    - id (int): Identificador único do usuário.
    - nome (varchar): Nome completo do usuário.
    - email (varchar): Endereço de e-mail do usuário.
    - endereco (varchar): Endereço residencial.
    - numero (int): Número do endereço.
    - documento (varchar): Documento de identificação (CPF/CNPJ).
    - senha (varchar): Senha de acesso ao sistema.

---

2. Fornecedores

- **Descrição**: Entidade que oferece créditos para os compradores (clientes).
- **Atributos**:
    - id (int): Identificador único do fornecedor.
    - razao_social (varchar): Nome empresarial do fornecedor.
    - telefone_comercial (varchar): Telefone de contato comercial.
    - uf (char[2]): Unidade federativa (estado) de atuação.

---

3. Clientes

- **Descrição**: Entidade que adquire créditos de fornecedores.
- **Atributos**:
    - id (int): Identificador único do cliente.
    - telefone_pessoal (varchar): Telefone de contato pessoal.

- descricao (text): Descrição ou observações adicionais sobre o cliente.

---

4. Ofertas

- **Descrição**: Entidade que representa os créditos negociados entre fornecedores e compradores.
- **Atributos**:
  - id (int): Identificador único da oferta.
  - valor_credito (decimal): Valor monetário do crédito.
  - data_emissao (date): Data de emissão da oferta.
  - data_vencimento (date): Data de vencimento do crédito.
  - id_fornecedor (int): Chave estrangeira referenciando o fornecedor.

---

5. Pagamento

- **Descrição**: Entidade que registra os pagamentos realizados pelos compradores para liquidar créditos adquiridos.
- **Atributos**:
  - id (int): Identificador único do pagamento.
  - id_compra (int): Chave estrangeira para o crédito pago.
  - Id_tipo_pagamento(int): Chave estrangeira para o tipo de pagamento.
  - valor_pago (decimal): Valor total pago.
  - data_pagamento (date): Data em que o pagamento foi realizado.
  - Num_titulo (int): Número do título.
  - Num_parcela (int): Número da parcela.

---

6. Vendas

- **Descrição**: Entidade que mantém o histórico completo de todas as transações, incluindo compras, vendas e pagamentos.
- **Atributos**:

- id_mercado (int): Identificador do mercado ou contexto da transação.
- id_credito (int): Referência ao crédito envolvido.
- quantidade (int): Quantidade de créditos transacionados.
- id_fornecedor (int): Fornecedor envolvido na transação.
- id_comprador (int): Cliente/comprador envolvido.
- data_transacao (date): Data da transação.
- valor_transacao (decimal): Valor financeiro da transação.

---

7. Tabela_Preco

- **Descrição**: Padroniza os preços e taxas por unidade federativa (UF).
- **Atributos**:
  - id (int): Identificador único do registro.
  - uf (varchar[2]): Unidade federativa.
  - valor (decimal): Valor padrão do crédito.
  - taxa (int): Taxa aplicada conforme a UF.

---

8. Tipo_Usuario

- **Descrição**: Registra os diferentes tipos de usuários do sistema.
- **Atributos**:
  - id (int): Identificador do tipo.
  - nome_tipo (varchar): Nome ou descrição do tipo de usuário (ex: Administrador, Cliente, Fornecedor).

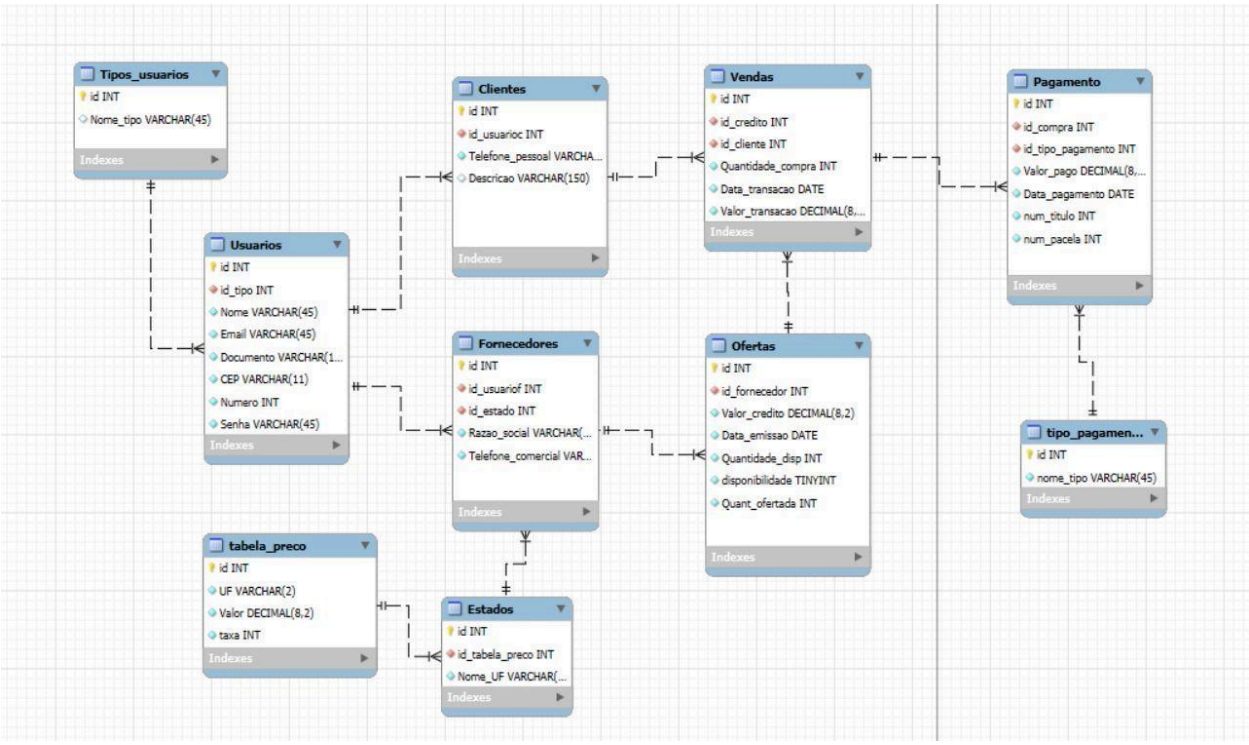
---

9. Tipo_Pagamento

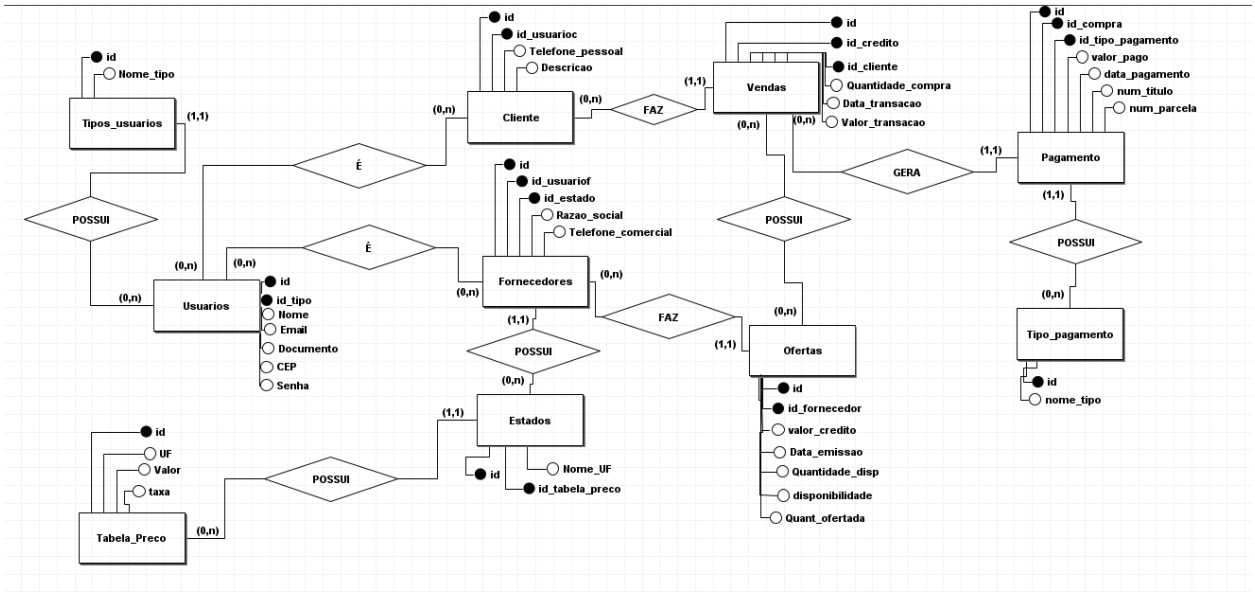
- **Descrição**: Registra os tipos de pagamentos disponíveis no sistema.
- **Atributos**:
  - id (int): Identificador do tipo de pagamento.

- nome_tipo (varchar): Descrição do tipo de pagamento (ex: Boleto, Cartão, PIX).

Modelo lógico:



Modelo Conceitual:



5 Considerações Finais

Obtivemos grande avanço referente ao banco de dados onde o mesmo se encontra totalmente alinhado com a idéia do projeto. Já a modelagem bpmn e uml, possuímos algumas dificuldades técnicas porém o acreditamos que o fluxo está claro sobre o processo do sistema. Já em programação orientada a objetos, realizamos um refinamentos nas classes, como invés de "Mercado", serem uma classe de "Compras" e "Vendas", e também já está sendo possível realizar algumas ações no console.

Referências Bibliográficas

[Elenque as referências bibliográficas utilizadas neste documento, seguindo as normas da ABNT e em ordem alfabética por sobrenome do primeiro autor.]

Projeto Final

Orientações:

Entregar este documento preenchido no **SchoolNet** até o prazo definido no cronograma do Projeto Interdisciplinar, com os requisitos solicitados inseridos no próprio documento e **Anexos** em arquivos separados.

1 Descrição da Proposta do Projeto

O BugigSolar é um ambiente online dinâmico e seguro, projetado para facilitar a negociação de créditos de energia solar. Usuários cadastrados podem se posicionar como Clientes, buscando e comprando créditos de forma eficiente, ou como Fornecedores, publicando e gerenciando suas ofertas. A plataforma simplifica todo o ciclo de transação, desde a publicação de ofertas até a geração de relatórios abrangentes de compras e vendas, promovendo um mercado transparente e acessível para energia renovável.

Requisitos do sistema:

1. Cadastro e Autenticação

- ***1.1. Cadastro de Usuários e Perfis:*** *O sistema deve permitir o cadastro de novos usuários, com a possibilidade de definirem-se como Cliente ou Fornecedor durante o processo de registro, associando seus dados básicos a um perfil específico.*
- ***1.2. Login de Usuários:*** *O sistema deve permitir que usuários registrados realizem login informando um e-mail e senha válidos.*
- ***1.3. Navegação por Perfil:*** *Após o login bem-sucedido, o sistema deve direcionar o usuário para um menu de funcionalidades específico do seu perfil (Cliente ou Fornecedor).*
- ***1.4. Encerramento de Sessão:*** *O sistema deve permitir que o usuário encerre sua interação com a aplicação, finalizando o programa.*

2. Funcionalidades para Fornecedores

- ***2.1. Publicação de Ofertas de Crédito:*** *O sistema deve permitir que o fornecedor publique novas ofertas de crédito, especificando a quantidade de créditos a serem disponibilizados para venda.*

- ***2.2. Preço de Crédito por UF:*** *Ao publicar uma oferta, o sistema deve associar o valor do crédito da oferta ao valor padrão definido para a Unidade Federativa (UF) do fornecedor, conforme registrado na tabela de estados.*
- ***2.3. Registro de Ofertas:*** *O sistema deve registrar a quantidade de créditos ofertada e a quantidade inicialmente disponível no momento da criação da oferta.*
- ***2.4. Finalização da Venda (Registro Pós-Transação):*** *O sistema deve registrar a transação de venda (compra do cliente e atualização da oferta) antes de registrar o pagamento do cliente.*

3. Funcionalidades para Clientes

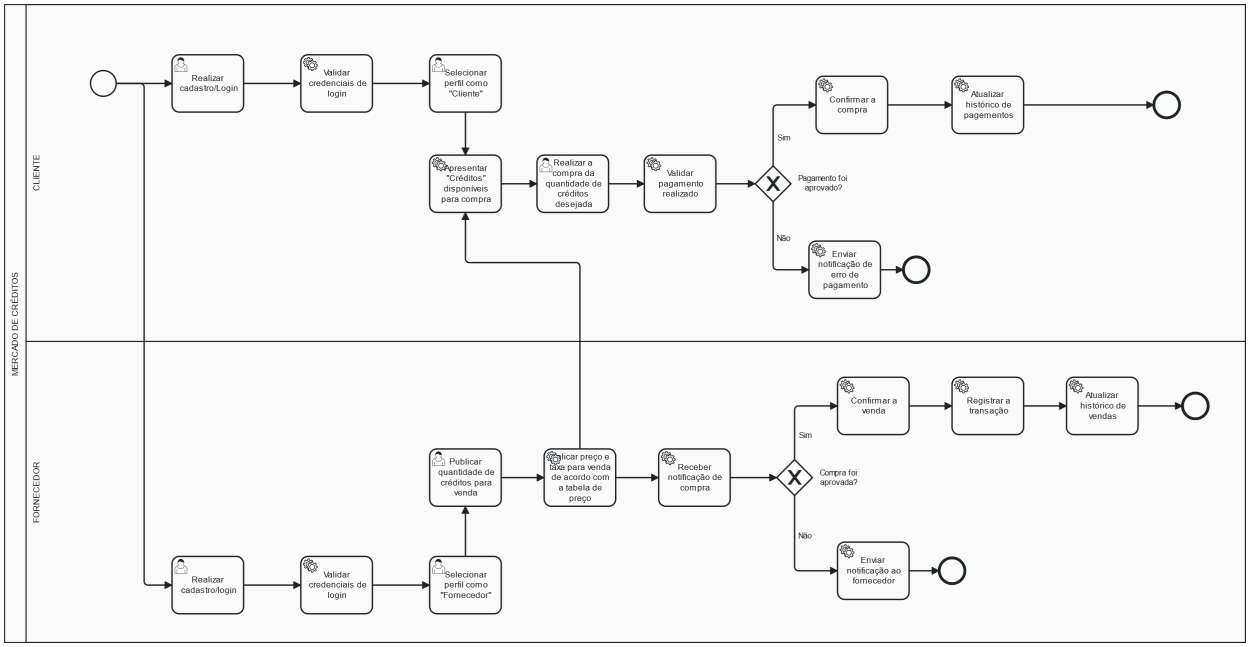
- ***3.1. Visualização e Seleção de Ofertas:*** *O sistema deve permitir que o cliente visualize as ofertas de crédito disponíveis para compra e selecione a oferta desejada, informando a quantidade que pretende adquirir.*
- ***3.2. Registro de Pagamento:*** *O sistema deve permitir que o cliente registre os detalhes do pagamento (tipo de pagamento, valor e data) após a conclusão da transação de compra.*
- ***3.3. Notificação de Registro de Pagamento:*** *O sistema deve informar ao cliente se o registro do pagamento foi bem-sucedido.*

4. Histórico e Relatórios

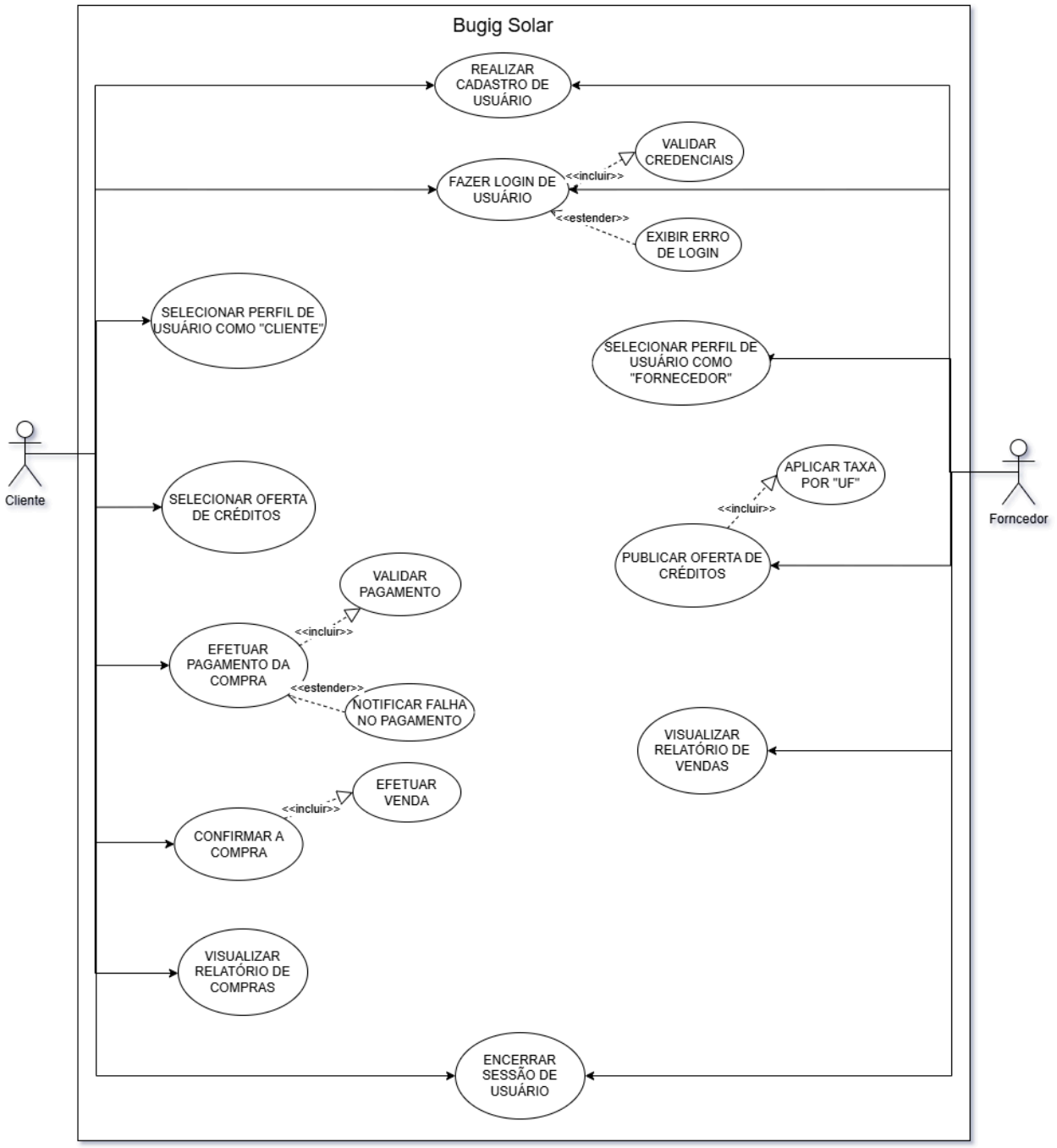
- ***4.1. Manutenção de Histórico de Transações:*** *O sistema deve manter um histórico detalhado de todas as transações de compra e dos pagamentos realizados na plataforma.*
- ***4.2. Registro de Vendas de Fornecedores:*** *O sistema deve registrar as ofertas publicadas por fornecedores e as compras realizadas pelos clientes a partir dessas ofertas.*
- ***4.3. Registro de Pagamentos de Clientes:*** *O sistema deve registrar todos os pagamentos efetuados pelos clientes.*
- ***4.4. Geração de Relatórios Básicos:*** *O sistema deve permitir a geração de relatórios que exibem um resumo das compras realizadas por clientes e das ofertas criadas por fornecedores.*

2 Modelagem de Negócios

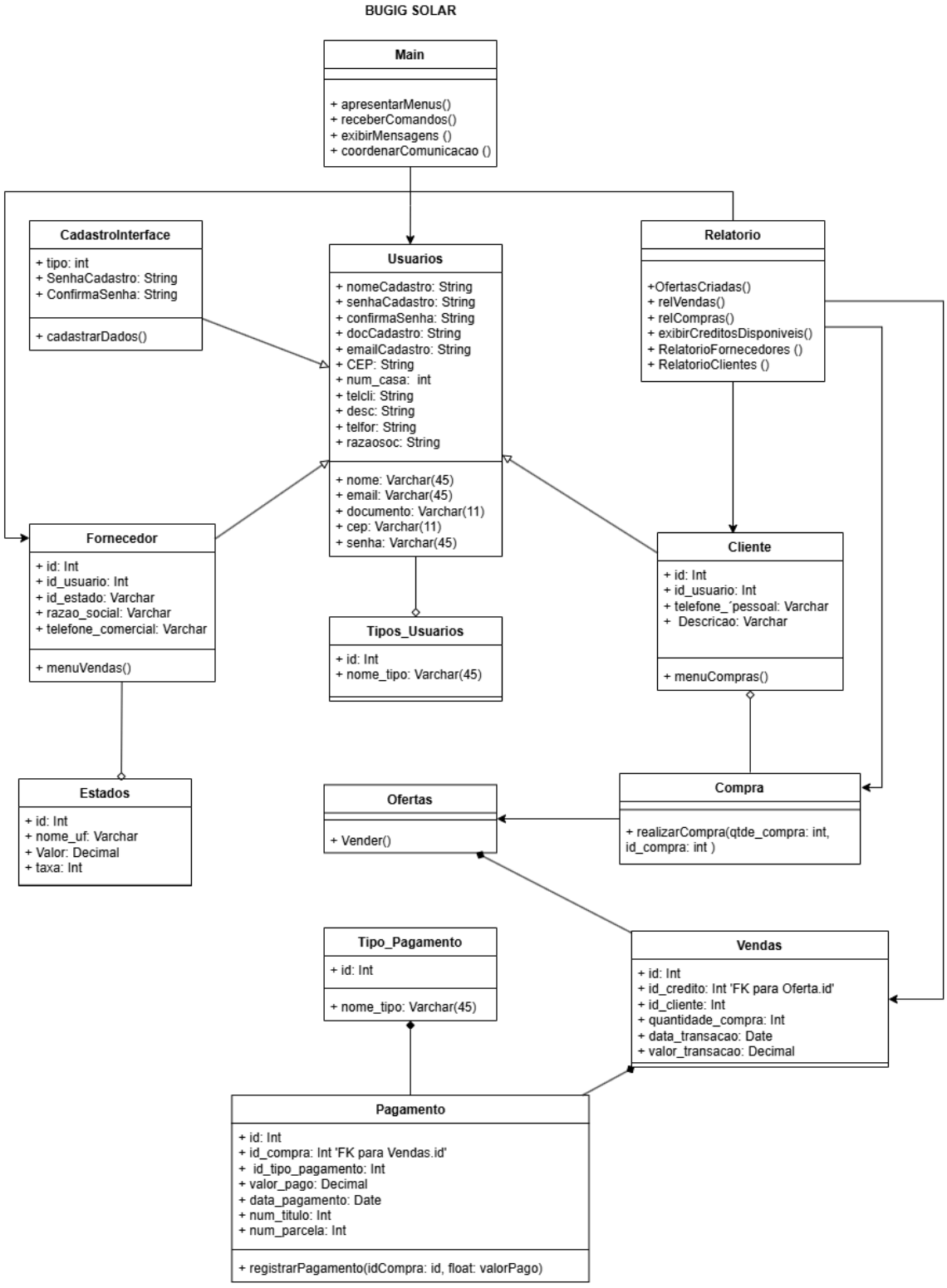
Modelagem dos Processos utilizando a técnica BPMN



Modelagem de Negócio UML: Caso de Uso e Atividades



Modelagem de Negócio UML: Diagrama de Classes



1. Herança (Generalização)

linha com seta triangular vazia:

- Cliente e Fornecedor **herdam de** Usuarios.
- CadastroInterface é uma **interface implementada por** Usuarios.

---

2. Composição

linha com losango preto

- Pagamento → Vendas: O pagamento não existe sem a venda.
- Vendas → Ofertas: A venda só ocorre se há uma oferta específica.
- Pagamento → Tipo_Pagamento

---

3. Agregação

Linha com losango branco:

- *Usuarios → Tipos_Usuarios: Um usuário tem um tipo, mas o tipo pode existir sozinho.*
- *Fornecedor → Estados: Um fornecedor está ligado a um estado, mas o estado pode existir sozinho.*
- *Compra → Cliente: Uma compra pertence a um cliente, mas o cliente existe fora da compra.*

---

4. Associação simples

linha com ponta de seta comum:

- *Compra → Ofertas: Compra se associa à oferta.*
- *Relatório → Vendas, Compras, Clientes, Fornecedores: São usados para gerar relatórios.*
- *Main → Usuarios: Main usa a classe Usuarios para realizar operações como login, cadastro e controle de acesso.*

3 Programação Orientada a Objetos

1. Classe Main

Responsabilidade: Gerenciar a interação inicial com o usuário por meio do terminal, apresentando as opções de login ou cadastro e direcionando para as funcionalidades adequadas.

Funcionalidades:

- Apresentar menus e opções disponíveis ao usuário (Login ou Cadastro).
- Receber entradas e comandos do usuário para seleção de opções.
- Exibir mensagens informativas e mensagens de erro (ex: "Valor Inválido!").
- Coordenar a comunicação entre as demais classes do sistema (CadastroDB, CadastroInterface, Cliente, Fornecedor).
- Inicializar a atualização de ofertas (Compra.atualizaOferta()).

Atributos:

- respostaLogCad: Armazena a resposta do usuário para as opções de Login ou Cadastro.
- email: Armazena o email inserido pelo usuário para login.
- senha: Armazena a senha inserida pelo usuário para login.

Métodos:

- main(String[] args): Ponto de entrada principal da aplicação, gerencia o fluxo inicial de login/cadastro.

---

2. Classe CadastroDB

Responsabilidade: Gerenciar todas as operações de persistência de dados relacionadas a usuários (clientes e fornecedores), estados e ofertas no banco de dados.

Funcionalidades:

- Selecionar o ID da UF de um fornecedor.
- Selecionar o ID de um fornecedor com base no email do usuário.
- Selecionar o ID de um cliente com base no email do usuário.
- Selecionar o ID de um usuário (genérico) com base no email. **(Novo)**

- Realizar o login de um usuário (cliente ou fornecedor) verificando email e senha.
- Cadastrar um novo usuário genérico na tabela usuarios.
- Cadastrar um novo cliente na tabela clientes.
- Cadastrar um novo fornecedor na tabela fornecedores, incluindo a UF e razão social.
- Obter o ID de um estado a partir de sua sigla.
- Obter o valor do crédito associado a uma UF específica.

Atributos:

- N/A (A classe utiliza variáveis locais dentro dos métodos, sem atributos de instância explícitos definidos no snippet).

Métodos:

- selUF(): Seleciona o ID da UF de um fornecedor.
- selFor(): Seleciona o ID de um fornecedor.
- selCli(): Seleciona o ID de um cliente.
- selID_user(String email): Seleciona o ID de um usuário genérico (cliente ou fornecedor) com base no email. **(Novo)**
- fazerLogin(String email, String senha): Realiza o login do usuário.
- cadastrarUser(): Cadastra um usuário genérico.
- cadastrarCli(): Cadastra um cliente.
- cadastrarFor(): Cadastra um fornecedor.
- getID_UF(): Obtém o ID de uma UF.
- getValorCredito(): Obtém o valor do crédito para uma UF.

---

3. Classe CadastroInterface

Responsabilidade: Gerenciar o processo de cadastro de novos usuários no sistema, coletando dados e validando-os antes de persistir no banco de dados.

Funcionalidades:

- Apresentar o menu de cadastro ao usuário.
- Coletar dados como tipo de usuário (Cliente ou Fornecedor), nome completo, e-mail, senha e confirmação de senha.
- Validar as entradas do usuário, como nome e correspondência de senhas.

- Para clientes, solicitar telefone pessoal e descrição.
- Para fornecedores, solicitar telefone comercial, UF de origem e razão social.
- Chamar métodos de CadastroDB para persistir os dados do usuário.

Atributos:

- tipo: Armazena o tipo de usuário selecionado (1 para Cliente, 2 para Fornecedor).
- senhaCadastro: Armazena a senha digitada durante o cadastro (usada para validação temporária).
- confirmaSenha: Armazena a confirmação da senha digitada durante o cadastro (usada para validação temporária).

Métodos:

- cadastrarDados(): Coleta e valida dados para cadastro de clientes ou fornecedores.
- cadastroCliente(): (Não explícito nos snippets, mas inferido como parte de cadastrarDados para lógica específica do cliente).
- cadastroFornecedor(): (Não explícito nos snippets, mas inferido como parte de cadastrarDados para lógica específica do fornecedor).

---

4. Classe Cliente

Responsabilidade: Fornecer e gerenciar o menu de opções específico para usuários do tipo cliente.

Funcionalidades:

- Apresentar o "Menu de Compras" ao cliente.
- Receber a escolha do cliente (Comprar, Relatórios, Sair do sistema).
- Direcionar para a funcionalidade de compra.
- Direcionar para a funcionalidade de relatórios específicos para clientes.
- Encerrar o sistema se o cliente optar por sair.
- Tratar opções inválidas inseridas pelo usuário.

Atributos:

- resposta: Armazena a opção escolhida pelo cliente no menu de compras.

Métodos:

- MenuCompras(): Exibe o menu de compras e gerencia as opções do cliente.

---

5. Classe Compra

Responsabilidade: Gerenciar todas as operações relacionadas à compra de créditos, incluindo a exibição de ofertas disponíveis, registro da compra e atualização do estoque.

Funcionalidades:

- Atualizar a disponibilidade de ofertas no banco de dados.
- Realizar uma compra, permitindo ao cliente selecionar ofertas com base na quantidade desejada.
- Atualizar o registro de compras no banco de dados.
- Registrar a compra e o valor pago em uma transação.
- Listar ofertas disponíveis para compra.
- Integrar com a funcionalidade de pagamento.

Atributos:

- N/A (A classe utiliza variáveis locais dentro dos métodos, sem atributos de instância explícitos definidos no snippet).

Métodos:

- updateOf(): Atualiza a disponibilidade de ofertas no banco de dados.
- realizaCompra(int qtd_compra, int id_compra): Registra uma compra e seu valor.
- atualizarCompra(int qtd_compra, int id_compra): Atualiza a quantidade de uma oferta após uma compra.
- atualizaOferta(): Atualiza o status das ofertas.
- comprar(): Gerencia o processo de compra de créditos pelo cliente.

---

6. Classe Conexao

Responsabilidade: Estabelecer e gerenciar a conexão com o banco de dados MySQL.

Funcionalidades:

- Fornecer uma conexão única e reutilizável com o banco de dados.
- Tratar exceções de conexão com o banco de dados.

Atributos:

- url: A URL de conexão com o banco de dados.
- user: O nome de usuário para acessar o banco de dados.
- password: A senha para acessar o banco de dados.
- conn: O objeto de conexão JDBC que armazena a conexão ativa.

Métodos:

- getConexao(): Retorna uma instância da conexão com o banco de dados.

---

7. Classe Fornecedor

Responsabilidade: Fornecer e gerenciar o menu de opções específico para usuários do tipo fornecedor.

Funcionalidades:

- Apresentar o "Menu de Vendas" ao fornecedor.
- Receber a escolha do fornecedor (Vender, Relatórios, Sair).
- Direcionar para a funcionalidade de criação de ofertas.
- Direcionar para a funcionalidade de relatórios específicos para fornecedores.
- Encerrar o sistema se o fornecedor optar por sair.
- Tratar opções inválidas inseridas pelo usuário.

Atributos:

- resposta: Armazena a opção escolhida pelo fornecedor no menu de vendas.

Métodos:

- MenuVendas(): Exibe o menu de vendas e gerencia as opções do fornecedor.

---

8. Classe Input

Responsabilidade: Fornecer uma instância única e global de Scanner para facilitar a leitura de entradas do usuário em todo o sistema.

Funcionalidades:

- Permitir que outras classes acessem um objeto Scanner pré-configurado para entrada de dados do console.

Atributos:

- teclado: Uma instância estática e final de Scanner para leitura de entrada do console.

Métodos:

- N/A (A classe é composta por um único atributo estático e final, sem métodos explícitos além do construtor padrão, que não é visível).

---

9. Classe Ofertas

Responsabilidade: Gerenciar as operações relacionadas à criação e registro de ofertas de venda de créditos por parte dos fornecedores.

Funcionalidades:

- Permitir que um fornecedor crie uma nova oferta de venda.
- Consultar o valor de crédito para a UF de origem do fornecedor.
- Persistir os detalhes da oferta (ID do fornecedor, valor do crédito, data de emissão, quantidade, disponibilidade) no banco de dados.
- Exibir mensagens de sucesso ou erro durante a criação da oferta.

Atributos:

- qtdvenda: Armazena a quantidade de créditos que o fornecedor deseja colocar à venda.

Métodos:

- Vender(): Gerencia o processo de criação de uma nova oferta de venda por parte do fornecedor.

---

10. Classe Pagamentos

Responsabilidade: Registrar informações de pagamento no banco de dados.

Funcionalidades:

- Registrar um pagamento associado a uma compra.
- Solicitar ao usuário a opção de pagamento (Crédito, Débito, Pix, Boleto).
- Persistir os detalhes do pagamento (ID da compra, tipo de pagamento, valor pago, data) no banco de dados.
- Exibir mensagens de sucesso ou erro durante o registro do pagamento.

Atributos:

- N/A (A classe utiliza variáveis locais dentro dos métodos, sem atributos de instância explícitos definidos no snippet).

Métodos:

- registrarPagamento(int idCompra, float valorPago): Registra os detalhes de um pagamento no banco de dados.

---

11. Classe Relatorios

Responsabilidade: Gerar e exibir diferentes tipos de relatórios para clientes e fornecedores, recuperando dados relevantes do banco de dados.

Funcionalidades:

- Exibir relatórios de ofertas criadas por um fornecedor.
- Exibir relatórios de compras realizadas por um cliente.
- Apresentar um menu de relatórios específico para fornecedores, com opções de ver ofertas criadas e voltar.
- Apresentar um menu de relatórios específico para clientes, com opções de ver relatórios de compras e voltar.
- Coordenar a navegação de volta aos menus principais.

Atributos:

- N/A (A classe utiliza variáveis locais dentro dos métodos, sem atributos de instância explícitos definidos no snippet).

Métodos:

- ofertasCriadas(): Exibe um relatório das ofertas criadas por um fornecedor.
- relCompras(): Exibe um relatório das compras realizadas por um cliente.
- relatorioFornecedores(): Exibe o menu de relatórios para fornecedores e gerencia suas opções.
- relatorioClientes(): Exibe o menu de relatórios para clientes e gerencia suas opções.

---

12. Classe Usuarios

Responsabilidade: Servir como um modelo de dados (POJO - Plain Old Java Object) para armazenar informações de usuários (clientes e fornecedores) e seus atributos, funcionando como um "container" para os dados manipulados pelas outras classes.

Funcionalidades:

- Armazenar e fornecer acesso (getters e setters) a diversos atributos relacionados a usuários, como nome, email, senhas, tipo de usuário, IDs (de usuário, cliente, fornecedor, UF), telefones, descrição e razão social.

Atributos:

- nomeCadastro: Nome completo do usuário.
- senhaCadastro: Senha do usuário.
- confirmaSenha: Confirmação da senha para validação.
- docCadastro: Documento de cadastro (não utilizado nos snippets fornecidos).
- emailCadastro: Email do usuário.
- CEP: CEP do endereço do usuário (não utilizado nos snippets fornecidos).
- num_casa: Número da casa do usuário (não utilizado nos snippets fornecidos).
- telcli: Telefone pessoal do cliente.
- desc: Descrição adicional do cliente.
- telfor: Telefone comercial do fornecedor.
- razaosoc: Razão social do fornecedor.
- tipouser: Tipo de usuário (1 para Cliente, 2 para Fornecedor).
- IDforn: ID do fornecedor.

- IDuser: ID do usuário na tabela usuarios.
- IDclient: ID do cliente.
- ID_UF: ID da Unidade Federativa.
- UF: Sigla da Unidade Federativa.

Métodos:

- getID_UF(): Retorna o ID da UF.
- setID_UF(int iD_UF): Define o ID da UF.
- getUF(): Retorna a sigla da UF.
- setUF(String uF): Define a sigla da UF.
- getIDforn(): Retorna o ID do fornecedor.
- setIDforn(int iDforn): Define o ID do fornecedor.
- getIDuser(): Retorna o ID do usuário.
- setIDuser(int iDuser): Define o ID do usuário.
- getIDclient(): Retorna o ID do cliente.
- setIDclient(int iDclient): Define o ID do cliente.
- getNomeCadastro(): Retorna o nome de cadastro.
- setNomeCadastro(String nomeCadastro): Define o nome de cadastro.
- getSenhaCadastro(): Retorna a senha de cadastro.
- setSenhaCadastro(String senhaCadastro): Define a senha de cadastro.
- getConfirmaSenha(): Retorna a confirmação da senha.
- setConfirmaSenha(String confirmaSenha): Define a confirmação da senha.
- getdocCadastro(): Retorna o documento de cadastro.
- setdocCadastro(String docCadastro): Define o documento de cadastro.
- getEmailCadastro(): Retorna o email de cadastro.
- setEmailCadastro(String emailCadastro): Define o email de cadastro.
- getCEP(): Retorna o CEP.
- setCEP(String cEP): Define o CEP.

- getNum_casa(): Retorna o número da casa.
- setNum_casa(int num_casa): Define o número da casa.
- getTelcli(): Retorna o telefone do cliente.
- setTelcli(String telcli): Define o telefone do cliente.
- getTelfor(): Retorna o telefone do fornecedor.
- setTelfor(String telfor): Define o telefone do fornecedor.
- getRazaosoc(): Retorna a razão social.
- setRazaosoc(String razaosoc): Define a razão social.
- getTipouser(): Retorna o tipo de usuário.
- setTipouser(int tipouser): Define o tipo de usuário.
- getDesc(): Retorna a descrição do cliente.
- setDesc(String desc): Define a descrição do cliente.

---

13. Classe Validacao

Responsabilidade: Fornecer métodos estáticos para validar diferentes tipos de entradas de dados, garantindo a integridade e o formato correto das informações inseridas pelo usuário.

Funcionalidades:

- Validar se um nome completo é válido (não nulo, não vazio, contém espaço, não contém dígitos).
- Validar um CPF (formato de 11 dígitos, não sequência de dígitos iguais, apenas números).
- Validar uma cidade (não nula, não vazia).
- Validar um estado (não nulo, não vazio, não contém dígitos).
- Validar um email (formato básico com '@' e '.').
- Validar um CEP (formato de 8 dígitos, apenas números).
- Validar uma senha (mínimo de 6 caracteres).

- Validar um CNPJ (formato de 14 dígitos, não sequência de dígitos iguais, apenas números).
- Validar o tipo de usuário (1 ou 2), solicitando nova entrada em caso de valor inválido.
- Obter e formatar a data e hora atual.

Atributos:

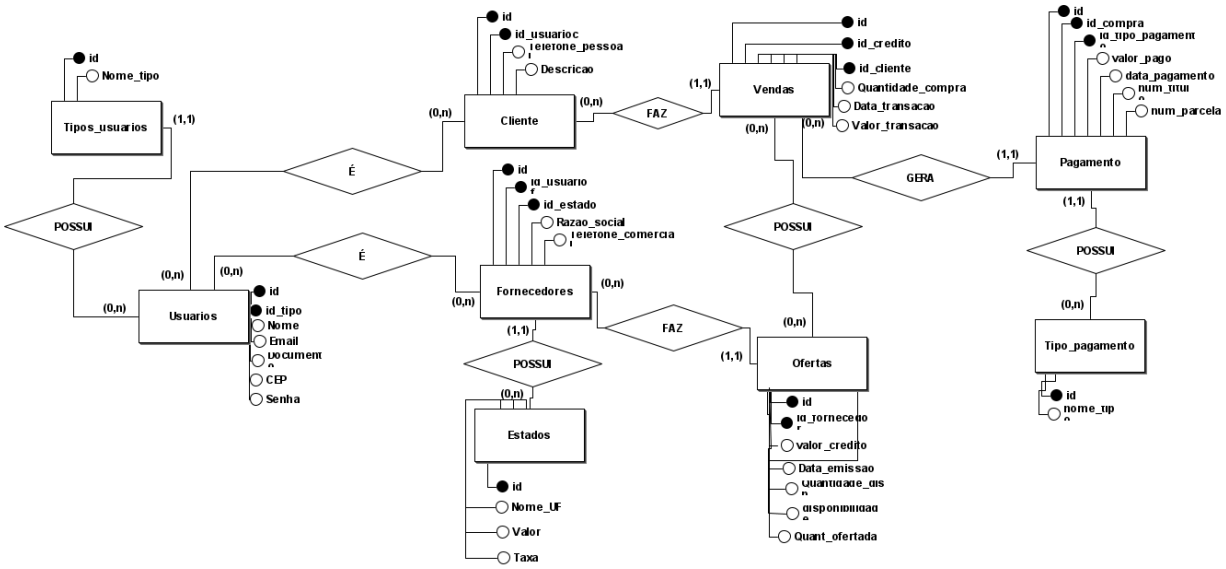
- N/A (A classe utiliza variáveis locais dentro dos métodos, sem atributos de instância explícitos definidos no snippet).

Métodos:

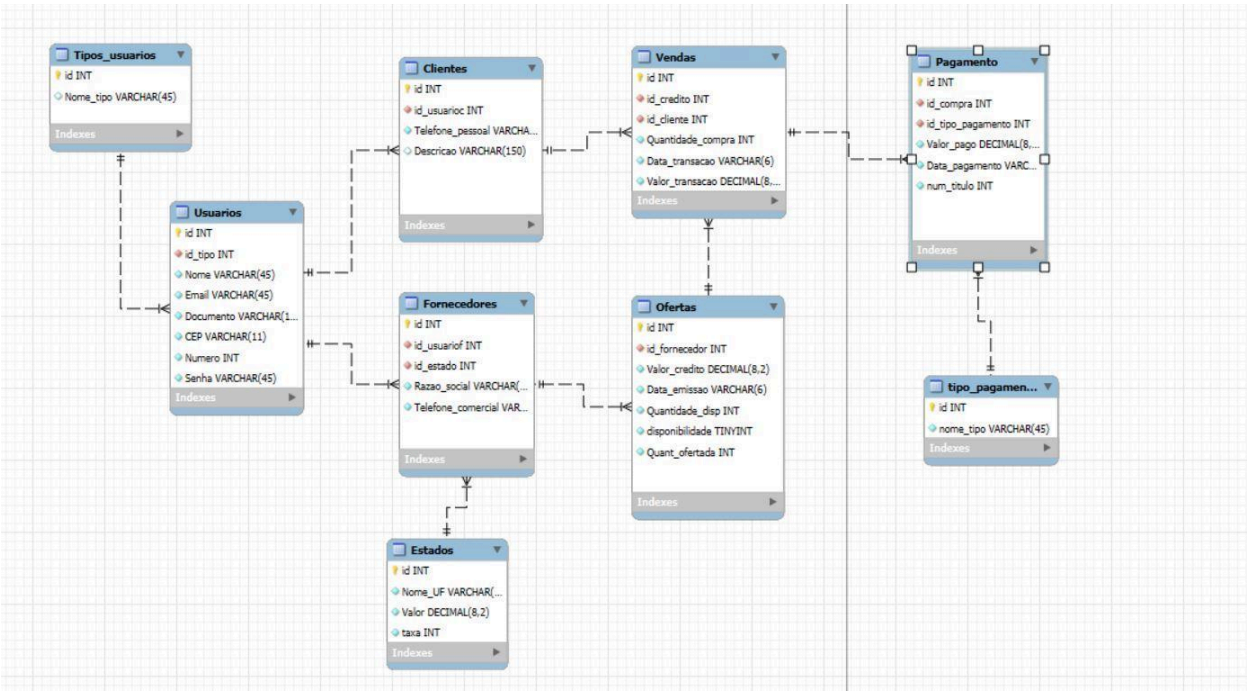
- validarNomeCompleto(String nome): Valida o formato de um nome completo.
- validarCPF(String cpf): Valida o formato de um CPF.
- validarCidade(String cidade): Valida se uma cidade é válida.
- validarEstado(String estado): Valida se um estado é válido.
- validarEmail(String email): Valida o formato de um email.
- validarCEP(String cep): Valida o formato de um CEP.
- validarSenha(String senha): Valida se uma senha atende aos requisitos.
- validarCNPJ(String cnpj): Valida o formato de um CNPJ.
- validarTipo(): Valida a entrada do tipo de usuário (cliente/fornecedor).
- setarData(): Retorna a data e hora atual formatada.

4 Banco de Dados

Modelo Conceitual de Banco de Dados



Modelo Lógico do Banco de Dados



1. Usuarios

Descrição: *Entidade central que armazena os dados cadastrais básicos de todos os usuários do sistema, sejam eles clientes ou fornecedores.*

Atributos:

- *id (INT): Identificador único e primário do usuário.*
- *Nome_usuario (VARCHAR): Nome completo do usuário.*
- *Email (VARCHAR): Endereço de e-mail único do usuário, utilizado para login.*
- *Endereco (VARCHAR): Endereço residencial ou comercial do usuário.*
- *Numero_end (INT): Número do imóvel no endereço do usuário.*
- *Documento (VARCHAR): Número do documento de identificação do usuário (CPF ou CNPJ).*
- *Senha (VARCHAR): Senha de acesso ao sistema, armazenada de forma segura (espera-se hash).*

---

2. Fornecedores

Descrição: *Entidade que representa as empresas ou indivíduos que atuam como fornecedores de créditos no sistema, associados a um usuário.*

Atributos:

- *id (INT): Identificador único e primário do fornecedor.*
- *id_usuario (INT): Chave estrangeira que referencia o id da tabela Usuarios, vinculando o fornecedor a um usuário.*
- *id_estado (INT): Chave estrangeira que referencia o id da tabela Estados, indicando a unidade federativa de atuação do fornecedor.*
- *Razao_social (VARCHAR): Nome empresarial ou social do fornecedor.*
- *Telefone_comercial (VARCHAR): Telefone de contato comercial do fornecedor.*

---

3. Clientes

Descrição: *Entidade que representa os usuários que adquirem créditos de fornecedores no sistema, associados a um usuário.*

Atributos:

- *id (INT): Identificador único e primário do cliente.*
- *id_usuario (INT): Chave estrangeira que referencia o id da tabela Usuarios, vinculando o cliente a um usuário.*
- *Telefone_pessoal (VARCHAR): Telefone de contato pessoal do cliente.*
- *Descricao (TEXT): Campo para observações ou informações adicionais sobre o cliente.*

---

4. Ofertas

Descrição: *Entidade que representa os créditos disponibilizados pelos fornecedores para venda, com detalhes sobre o valor, quantidade e status de disponibilidade.*

Atributos:

- *id (INT): Identificador único e primário da oferta.*
- *id_fornecedor (INT): Chave estrangeira que referencia o id da tabela Fornecedores, indicando quem criou a oferta.*
- *Valor_credito (DECIMAL): Valor monetário de cada unidade de crédito na oferta.*
- *Data_emissao (DATE): Data em que a oferta foi criada e disponibilizada.*
- *Quantidade_ofertada (INT): Quantidade total de créditos inicialmente colocada à venda nesta oferta.*
- *Disponibilidade (INT): Indica se a oferta está ativa (1) ou inativa (0).*
- *Quantidade_disp (INT): Quantidade de créditos ainda disponível para compra nesta oferta.*

---

5. Pagamento

Descrição: *Entidade que registra os pagamentos efetuados pelos clientes para liquidar os valores referentes aos créditos adquiridos.*

Atributos:

- *id (INT): Identificador único e primário do registro de pagamento.*
- *id_compra (INT): Chave estrangeira que referencia o id da tabela Compras_Clientes, vinculando o pagamento à compra correspondente.*

- *id_tipo_pagamento (INT): Chave estrangeira que referencia o id da tabela Tipo_Pagamento, indicando o método de pagamento utilizado.*
- *Valor_pago (DECIMAL): Valor total pago nesta transação.*
- *Data_pagamento (DATE): Data em que o pagamento foi registrado.*
- *Num_titulo (INT): Número de referência do título (se aplicável, ex: boleto).*
- *Num_parcela (INT): Número da parcela (se o pagamento for parcelado).*

---

6. Vendas

Descrição: *Entidade que mantém o histórico completo das transações de compra de créditos pelos clientes, detalhando a oferta, fornecedor, cliente e valores envolvidos.*

Atributos:

- *id (INT): Identificador único e primário da transação de compra do cliente.*
- *id_credito (INT): Chave estrangeira que referencia o id da tabela Ofertas, indicando a oferta que foi comprada.*
- *Quantidade_comprada (INT): Quantidade de créditos adquirida nesta transação.*
- *id_fornecedor (INT): Chave estrangeira que referencia o id da tabela Fornecedores, indicando o fornecedor da oferta.*
- *id_cliente (INT): Chave estrangeira que referencia o id da tabela Clientes, indicando o cliente que realizou a compra.*
- *Data_transacao (DATE): Data em que a transação de compra foi realizada.*
- *Valor_transacao(DECIMAL): Valor financeiro total da transação de compra.*

---

7. Estados

Descrição: *Entidade que armazena informações sobre as unidades federativas (estados), incluindo um valor padrão de crédito e uma taxa associada a cada uma.*

Atributos:

- *id (INT): Identificador único e primário do estado.*
- *Nome_uf (VARCHAR[2]): Sigla da Unidade Federativa (ex: SP, RJ, MG).*

- *Valor (DECIMAL): Valor padrão do crédito associado a esta UF.*
- *Taxa (INT): Taxa percentual aplicada aos créditos desta UF (%).*

---

8. Tipo_Usuario

Descrição: *Entidade de lookup que categoriza os diferentes tipos de usuários permitidos no sistema (ex: Cliente, Fornecedor).*

Atributos:

- *id (INT): Identificador único e primário do tipo de usuário.*
- *Nome_tipo (VARCHAR): Nome ou descrição do tipo de usuário.*

---

9. Tipo_Pagamento

Descrição: *Entidade de lookup que registra os diferentes métodos de pagamento que podem ser utilizados nas transações do sistema (ex: Boleto, Cartão, PIX).*

Atributos:

- *id (INT): Identificador único e primário do tipo de pagamento.*
- *Nome_tipo (VARCHAR): Descrição do método de pagamento.*

5 Considerações Finais

Resultados Obtidos

O projeto BugigSolar atingiu seus objetivos iniciais, permitindo a simulação de um sistema de comercialização de créditos de energia solar. Foram desenvolvidas funcionalidades como cadastro de usuários, simulação de compra e venda de créditos, controle de transações e geração de relatórios. A estruturação com base em Programação Orientada a Objetos (POO) e banco de dados proporcionou uma base sólida e organizada para futuras evoluções.

Dificuldades e Lições Aprendidas

Entre as principais dificuldades enfrentadas, destacam-se a definição clara das regras de negócio e a modelagem correta dos relacionamentos entre entidades no banco de dados. Também houve desafios na aplicação prática dos diagramas UML e BPMN no desenvolvimento do sistema. Como lições aprendidas, reforçamos a importância da modelagem prévia, da organização do código e da clareza nos requisitos funcionais.

Contribuições Técnicas para a Comunidade Acadêmica

O projeto contribui tecnicamente ao demonstrar, na prática, como aplicar conceitos de Programação Orientada a Objetos, modelagem UML e lógica de banco de dados em um cenário realista de negócios. Essa experiência pode servir como referência para outros estudantes que buscam entender como essas disciplinas se integram no desenvolvimento de sistemas.

Contribuições para a Comunidade Acadêmica e Sociedade

O BugigSolar propõe uma solução que estimula o uso consciente de energia renovável e a valorização de recursos excedentes. Embora ainda em nível acadêmico, o conceito pode ser expandido para incentivar práticas sustentáveis e colaborar com a economia de energia em comunidades e instituições.

Trabalhos Futuros

Como sugestões para evoluções futuras do projeto, destacam-se:

- Criação de uma interface gráfica amigável;
- Implementação em ambiente web ou mobile;
- Simulação de notificações e feedback entre usuários

Referências Bibliográficas

PORTAL SOLAR. Créditos de energia solar: como funciona o sistema de compensação de energia. Disponível em: https://www.portalsolar.com.br/creditos-energia-solar. Acesso em: 19 de março de 2025.

Anexos

trabalho_pi_completo.zip

Apêndice A - Link do projeto no GitHub

Disponível em: https://github.com/HarryFelipe/Projeto_interdisciplinar_BugigSolar.git

Acesso em: 23 Junho 2025.

Apêndice A - Link de apresentação do projeto BugigSolar no Youtube

Disponível em: https://youtu.be/Mju-Sd2GyDM?feature=shared

Acesso em: 23 Junho 2025.

Anexo A - Matéria Publicada no site da FHO

Disponível em: https://www.fho.edu.br/noticia/3958

Acesso em: 23 junho 2025.

Anexo B - Foto da Equipe (Harry Felipe Santos Souza, Enzo Correa Caetano, Felipe Claudiano da Silva, Lucas Eduardo de Campos, Renan dos Santos Alvares.)

Disponivel em: https://www.fho.edu.br/noticia/3958

Acesso em: 05 junho 2025.